NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UNMANNED SYSTEMS: A LAB-BASED ROBOTIC ARM FOR GRASPING

by

Arturo Jacinto II

June 2015

Thesis Advisor:	Richard M. Harkins
Second Reader:	Peter Crooker

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2015	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE UNMANNED SYSTEMS: A LAB-BASED ROBOTIC ARM FOR GRASPING			**5. FUNDING NUMBERS**
6. AUTHOR(S) Arturo Jacinto II			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis implements the development of a Robotic Manipulation Laboratory to explore learning opportunities for various student experiments including the initial selection, startup and development of the Robotic arm and glove controller. The Robotic Manipulation Laboratory consists of a 6 Degree of Freedom robotic arm and a resistive glove controller that allows students to achieve hands-on understanding of the physics required to fabricate and maneuver a robotic arm. The Kinova JACO robotic arm was selected for its smooth operation, the ability to alter operational speed and open source programming examples. We chose a glove controller for ease of training and human-like efficiency. Testing on the JACO arm was completed. The JACO software was installed and verified for standalone operations within its range of motion. Tests on glove/arm interaction in the Robotic Operating System were completed and proved ineffective. Experiments with flex sensors on the glove for normal hand movements were completed and were successful.

14. SUBJECT TERMS robotic arm, glove controller, dynamics, DH parameters			**15. NUMBER OF PAGES** 49
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UNMANNED SYSTEMS: A LAB-BASED ROBOTIC ARM FOR GRASPING

Arturo Jacinto II
Lieutenant, United States Navy
B.S., Texas A&M University, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2015

Author: Arturo Jacinto II

Approved by: Richard M. Harkins
Thesis Advisor

Peter Crooker
Second Reader

Andres Larraza
Chair, Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis implements the development of a robotic manipulation laboratory to explore learning opportunities for various student experiments, including the initial selection, startup and development of the robotic arm and glove controller. The robotic manipulation laboratory consists of a 6 degree of freedom robotic arm and a resistive glove controller that allows students to achieve hands-on understanding of the physics required to fabricate and maneuver a robotic arm. The Kinova JACO robotic arm was selected for its smooth operation, the ability to alter operational speed, and open source programming examples. We chose a glove controller for ease of training and human-like efficiency. Testing on the JACO arm was completed. The JACO software was installed and verified for standalone operations within its range of motion. Tests on glove/arm interaction in the Robotic Operating System were completed and proved ineffective. Experiments with flex sensors on the glove for normal hand movements were completed and were successful.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AHRS	Attitude and Heading Reference System
AXV	Advanced experimental vehicle
DH	Denavit-Hartenberg
DIY	Do it yourself
DOF	Degrees of freedom
ICSP	In-circuit serial programming
IED	Improvised explosive device
IMU	Inertial measurement unit
PWM	Pulse width modulation
RML	robotic manipulation laboratory
ROS	Robot operating system
SCL	Serial Clock Line
SDA	Serial Data Line
USB	Universal serial bus

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor and second reader for all your support. I would also like to thank Steve Jacobs for his technical guidance during wiring and soldering of the glove controller.

I would like to acknowledge my wife for her support through this learning experience and for my entire military career. And to my kids, thank you for keeping the noise down when Daddy needed to focus on his thesis. I hope this part of your lives inspires your imagination and expands your awareness in the sciences.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Naval Postgraduate School Physics Department has implemented a robotic manipulation laboratory (RML) to explore learning opportunities for various student experiments. The focus of this project is to understand and apply the kinematics and dynamics for a 6 degree of freedom (DOF) Kinova JACO robotic arm, in a controlled lab environment, for various predetermined tasks under glove sensor control.

A. RESEARCH FOCUS

The JACO arm is a sophisticated and complex manipulator. End effector, the hand at the end of manipulator, trajectory and path planning are operationally and mathematically challenging for the 6 DOF arm. It was therefore deemed necessary to implement the manipulator in three phases: (1) Startup and Development, (2) Lab Implementation, (3) and Operational Demonstrations

This research project centered on Phase 1, while focusing on three subareas:

- JACO Manipulator Set Up And Integration
- Robot Operating System (ROS) Controller Development
- TeleOp Communication Infrastructure

The overarching goal for these focus areas was to integrate the JACO arm into an efficient, real-time and humanlike system where the trajectory motion emulates natural human arm movement. Lab demonstrations will show manipulator interaction under glove control. An operator will control the manipulator with the glove controller, which is accomplished by his or her respective computer communicating via the Internet and video feedback, as illustrated in Figure 1.

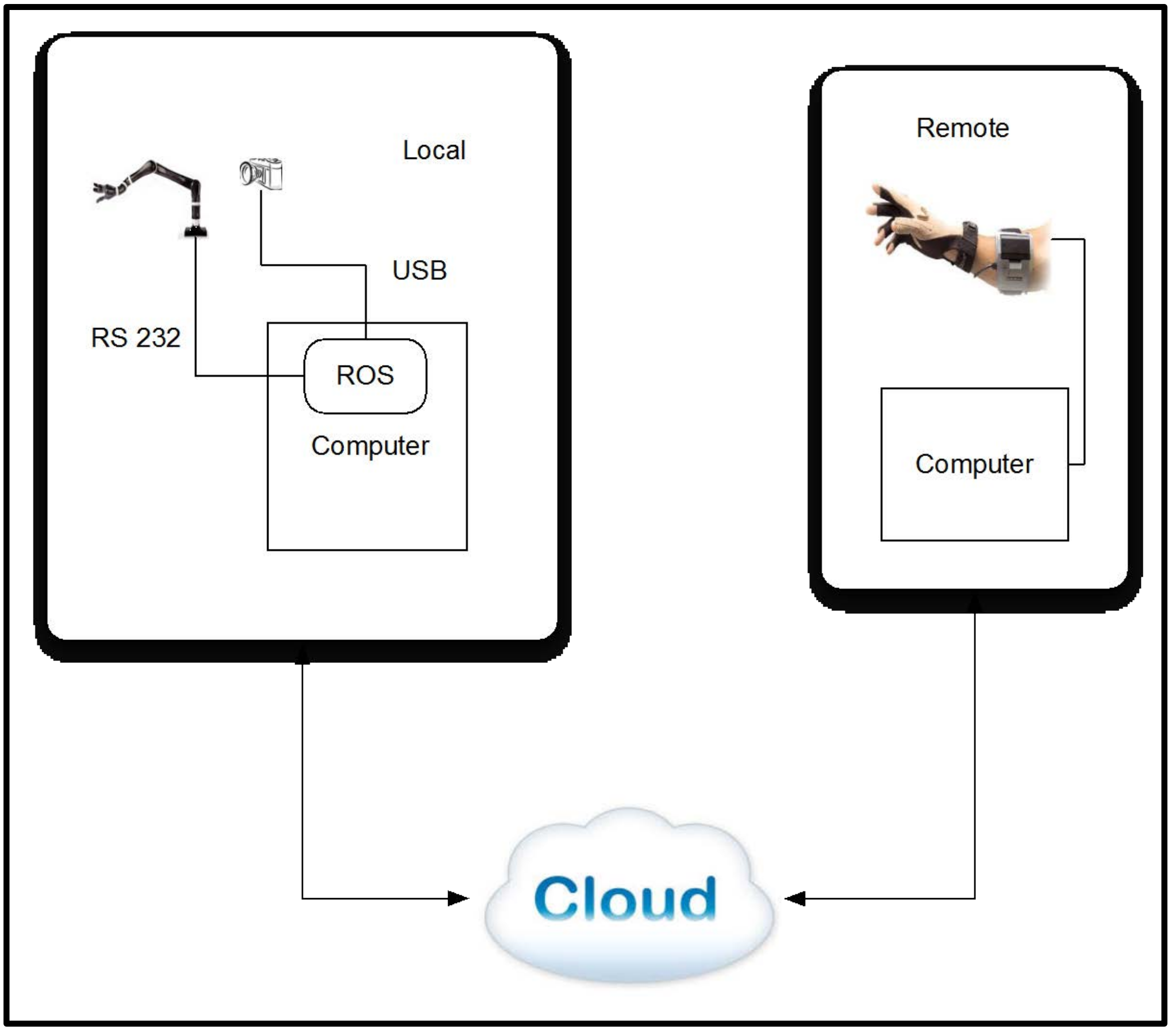


Figure 1. Overall system containing robotic arm, controller and communication infrastructure. An operator will control the manipulator with the glove controller, with their respective computers communicating via the Internet and video feedback.

B. EXPECTATIONS

The expectations for Phase 1 are to have selected a robotic arm, design a controller and establish a communications infrastructure. I will address the following questions:

- Do we need to build a robotic arm from scratch? If not, what is the best robotic arm on the market that fits our parameters?
- How do we make a controller that is efficient and quickly learned?
- How can we improve on our design?

II. THEORY

The fundamental physics used for the RML include Euler angles, dynamics, and the use of Denavit-Hartenberg (DH) parameters. This chapter will briefly explain the incorporation of these methods in the system.

A. EULER ANGLES

The controller needs to be able to produce an output for the manipulator to emulate. The IMU provides raw data via the accelerometer and magnetometer based on the position of the user's hand. The microcontroller reads the raw data and converts the data to Euler angles, in the **getOrientation** function, to generate the three-dimensional orientation data [1].

Euler angles allow us to express the orientation of any reference frame as an arrangement of three elemental rotations correlated to a known standard orientation, represented by another frame [2]. There are several different interpretations of Euler angles; however, the geometrical definition is easiest to understand. The geometrical definition centers on the axes of the original and rotated reference frames and an additional axis called the line of nodes [2]. The line of nodes is the intersection of the xy and the XY coordinate planes seen in Figure 2 [2].

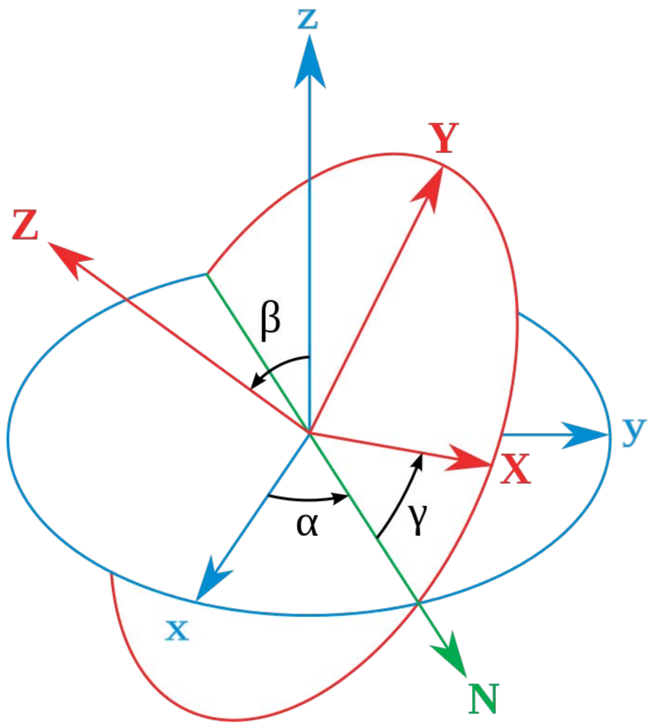


Figure 2. Euler angles depicting rotations about *z*, *N*, and *Z* axes with xyz (original) system in blue, XYZ (rotated) system in red and the line of nodes (N) in green from [2].

The **getOrientation** function uses the raw data from the IMU and inserts it into Equations 1, 2 and 3 from [1]. The roll (ψ), pitch (θ) and heading (ϕ) are calculated with the variables from Figure 2. A positive roll angle is the clockwise rotation about the positive X-axis, a positive pitch angle is the clockwise rotation about the positive Y-axis and a positive heading angle is the clockwise rotation about the positive Z-axis [2].

$$\text{roll: } \psi = \arctan 2\left(\frac{y}{z}\right) \tag{1}$$

$$\text{pitch: } \theta = \arctan\left(\frac{-x}{y\sin(\text{roll}) + z\cos(\text{roll})}\right) \tag{2}$$

$$\text{heading: } \phi = \arctan 2\left(\frac{z\sin(\text{roll}) - y\cos(\text{roll})}{x\cos(\text{pitch}) + y\sin(\text{pitch})\sin(\text{roll}) + z\sin(\text{pitch})\cos(\text{roll})}\right) \tag{3}$$

B. DYNAMICS AND DH PARAMETERS

Dynamics is the study of forces and their effect on motion [3]. Many factors have to be taken into account when finding the trajectory a manipulator travels. *Principles of Robot Motion* states, “A trajectory is a system dynamics problem that requires knowledge of the masses and inertias of the system, actuator limits and forces such as gravity and friction” [4]. Kinova simplified the work for the JACO arm with their proprietary algorithms and transformations from DH parameters to actual JACO physical angles.

Using Lagrangian dynamics for a manipulator allows for the use of computer algorithms for calculating equations of motion [4]. Subtracting the potential energy from the kinetic energy yields the Lagrangian of any mechanical system seen in Equation 4. L is the Lagrangian, K is the kinetic energy, V is the potential energy and q is a vector of generalized coordinates:

$$L(q,\dot{q}) = K(q,\dot{q}) - V(q) \tag{4}$$

The Lagrangian equations of motion can be derived via Equation 5 where u is a generalized force:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = u \tag{5}$$

Once all forces are found for the configuration of the system, then the forces can be rewritten into a 2nd order differential equation as in Equation 6. $M(q)$ is the inertial matrix, $C(q,\dot{q})\dot{q}$ is a vector of velocity products and $g(q)$ is a vector of gravitational forces.

$$u = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{6}$$

With the Lagrangian equation of motion, calculating planar rotation and spatial rotation for a rigid body is possible. Applying the kinetic energy of the system to Lagrange’s equation yields the torque about the chosen axis. The appeal of Lagrange’s equation is that it uses the fewest possible numbers to represent the orientation; however the equations are complicated [4].

The use of DH parameters is required to simplify the mathematics necessary to move the end effector of the manipulator to a point in space. The DH parameters do not include the hand, only the links. DH parameters are assigned to each link in a manipulator and are labeled θ_i, a_i, d_i and α_i [5]. a_i and α_i are the link length and link twist, respectively, and since they are determined by the construction of the links they are constants [5]. The only time a_i is a variable is if the link is attached to a prismatic joint. d_i and θ_i are the link offset and joint angle, respectively, and establish the location of the connected link [5]. d_i is constant and θ_i is the variable for a revolute joint [5].

To relate the end effector to the base of the manipulator, using homogeneous transformations, we have to establish reference frames for each link and correlate them to each other. Section 8.6.2 of [5] lists three rules for establishing link coordinate frames to get Figure 3.

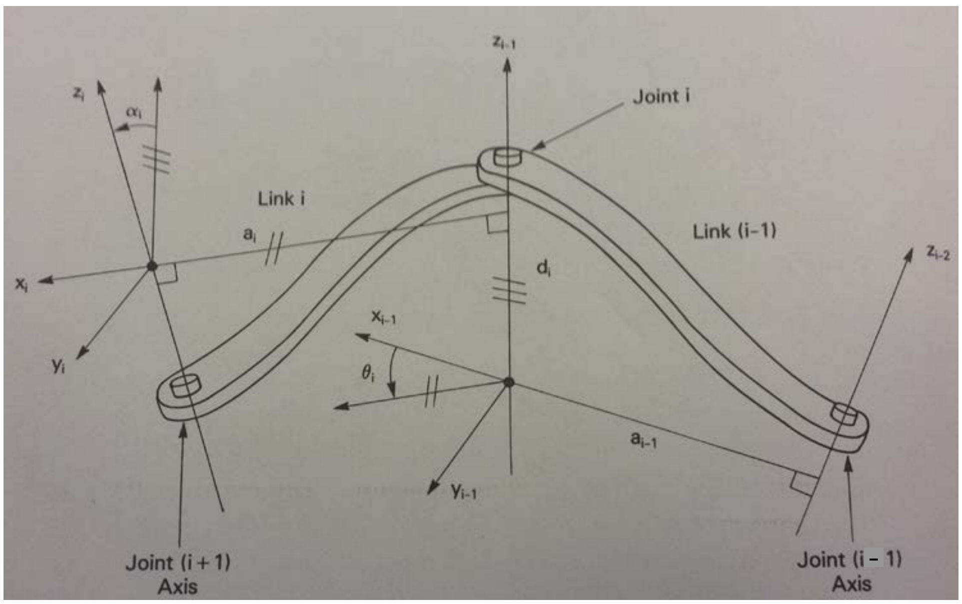


Figure 3. Coordinate frames for links and joints to use in the DH transformationfrom [5]

Using Figure 3 to relate each successive link coordinate frames, we can develop a DH matrix; see Equation 7 [5].

$$A_{(i-1),i} = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where yellow is the rotation matrix, red is the position vector, green is the perspective transform and orange is the scale factor. The first column of the rotation matrix is called the normal vector and makes up the x'-axis unit vector of reference frame 2 in terms of reference frame 1 [5]. The second and third columns of the rotation matrix make up the orientation vector (y'-axis unit vector) and approach vector (z'-axis unit vector), respectively, of reference frame 2 [5]. The position vector tells us where the origin of reference frame 2 is relative to reference frame 1 [5].

It is now possible to find a forward solution for a 6DOF manipulator to relate the end effector to the base of the manipulator. For an n-DOF manipulator, the expression for the DH operator is given by [5]:

$$T = A(n-1)n\ldots \tag{8}$$

therefore when $n = 6$,

$$T = A_{01}A_{12}A_{23}A_{34}A_{45}A_{56} \tag{9}$$

where the DH operator is the product of each DH matrix for each link. The final position of the end effector can be found by operating the DH operator on the initial end effector position, see Equation 10.

$$\text{Final Position} = T\begin{bmatrix} \mathrm{P}_x & \mathrm{P}_y & \mathrm{P}_z & 1 \end{bmatrix}^{\mathrm{T}} \tag{10}$$

THIS PAGE INTENTIONALLY LEFT BLANK

III. EQUIPMENT AND LAB SETUP

In this chapter, we describe the Kinova JACO Arm, the glove controller and the communications infrastructure.

A. THE JACO ARM

The JACO manipulator, see Figure 4, closely mimics the motion of the human arm. It has 6 DOF with a maximum reach of 90 cm and a maximum linear speed of 20 cm/s [6]. The arm weighs 5.3 kg and can lift a maximum payload of 2.5 kg at mid-range and 1.5 kg at full extension [6]. Kinova provides ROS open-source code, to ease the transition from joystick control to tele-optic control.

Figure 4. KINOVA JACO 6 DOF robotic arm attached to rolling table simulating lifting a bottle.

The JACO arm also comes with standalone software called JACOSOFT. The software allows the operator to *see* general information including:

- Arm trajectory, Figure 5
- Arm health, Figure 6

It also allows the operator to *change* operational configurations:

- Right-handed or left-handed preference
- Sensitivity
- Speed
- Protection zones

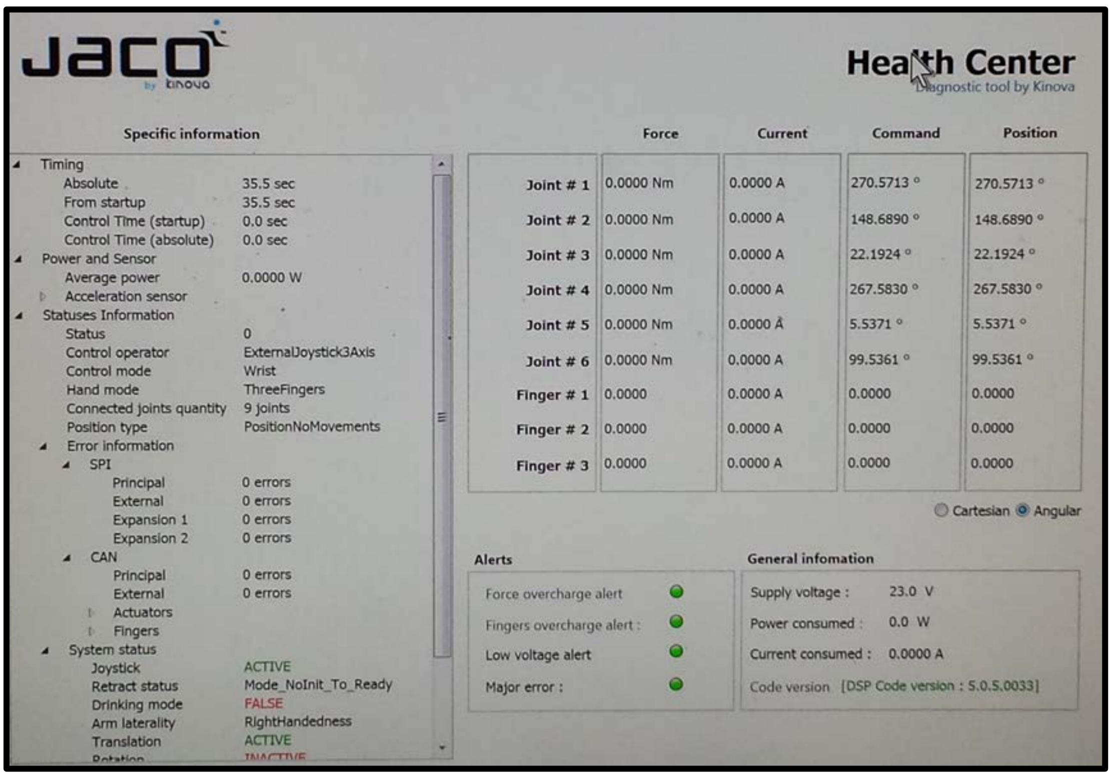


Figure 5. JACOSOFT Health Centerused to monitor JACO arm parameters. Force, current, command, and position of each joint and fingers are monitored to ensure the arm is operating normally and as desired.

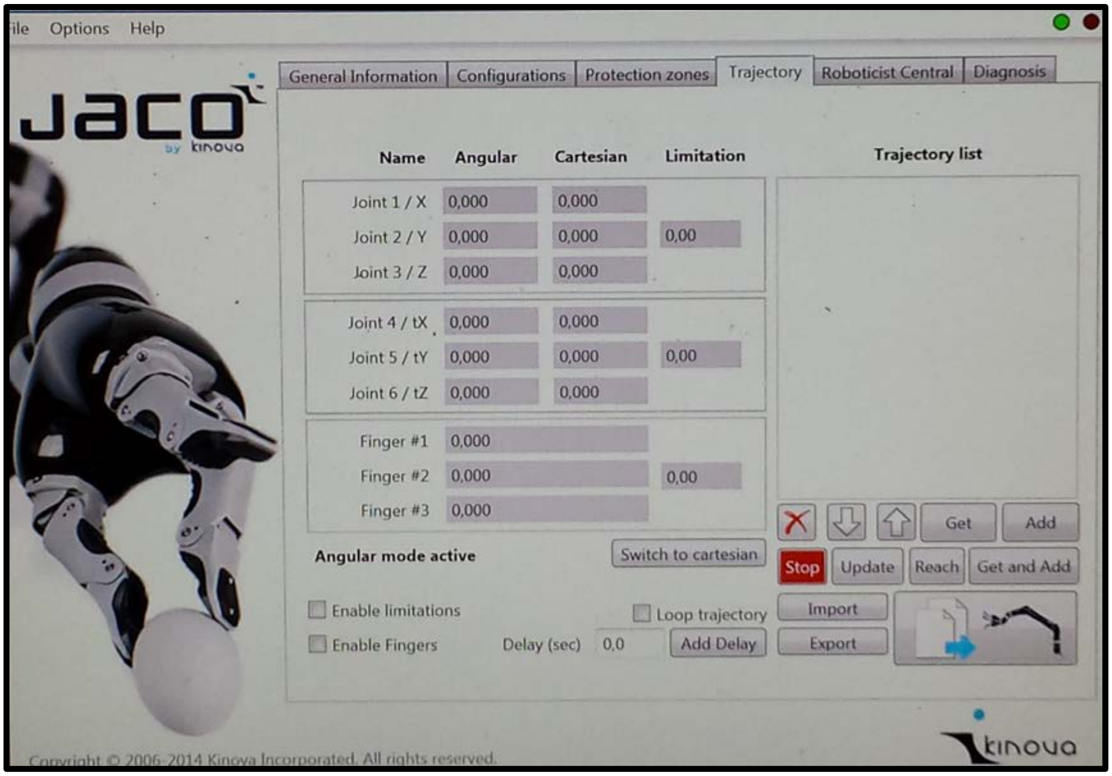


Figure 6. JACOSOFT trajectory pageused to monitor JACO arm motion during operation. An operator can get points along a path the manipulator is moving and saved as a trajectory to run at a later time.

JACOSOFT prohibits program code modifications for arm operation; however users can store path trajectories for various operations for future use. An example would be picking up a cup of water and returning it to its original position. JACOSOFT allows users to get comfortable controlling the arm.

The JACO arm was bolted to a rolling table for lab flexibility, see Figure 7. We added an electric bus so that we could power the arm and associated ancillary equipment including computer, network and camera from one location.

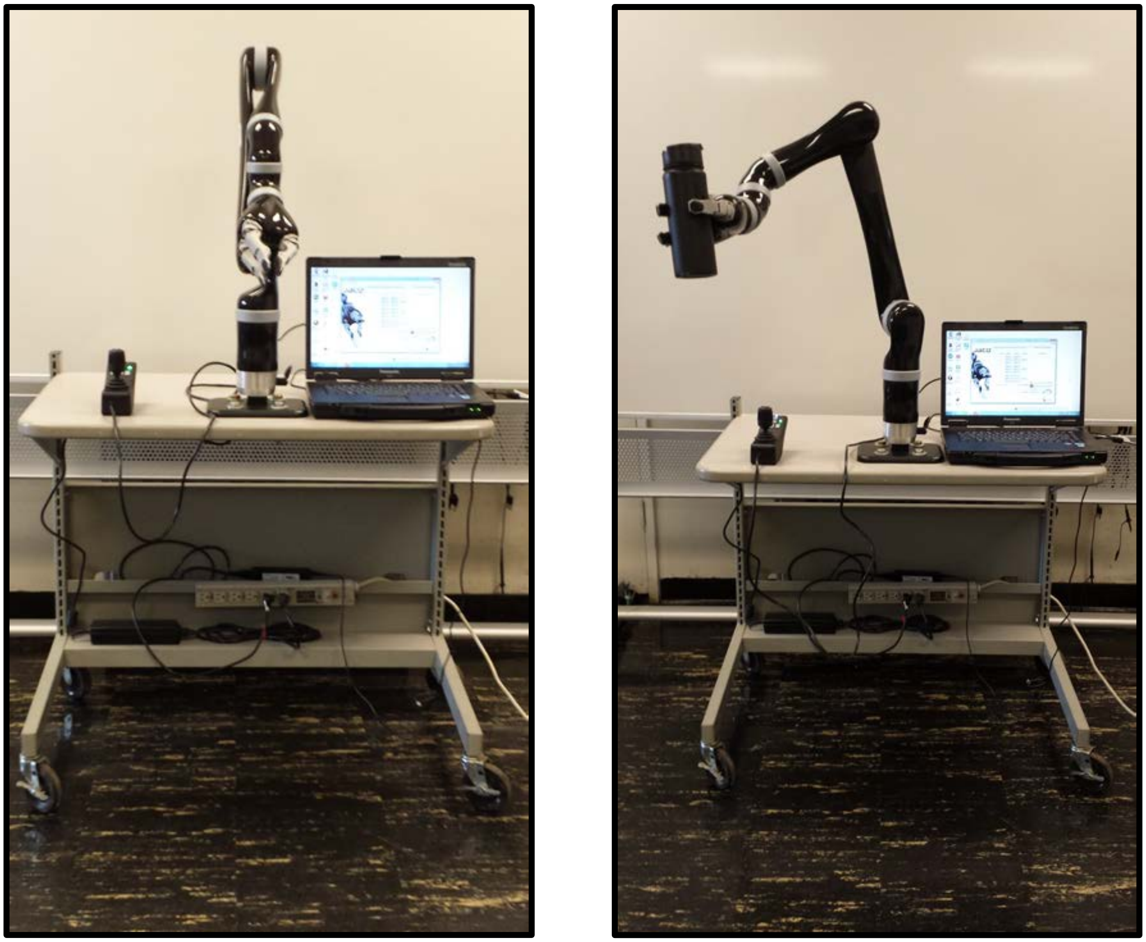


Figure 7. JACO Arm attached to rolling table for flexibility to move around the campus as needed and to simulate controlling the arm if attached to a moving vehicle.

B. CONTROLLER

As a replacement for the joystick, a resistive glove controller was implemented to invoke tele-optic control via natural hand movement. With the glove on a user's hand, the user moves the manipulator up/down, forward/backward and rotates it with regular hand motion. To invoke grasping action, we fitted the glove with flexible resistive sensors shown in Figure 8. The glove was modeled after prior work cited in open source literature in [7]. The glove consist of three main components; the IMU, microcontroller and flex sensors shown in Figure 8.

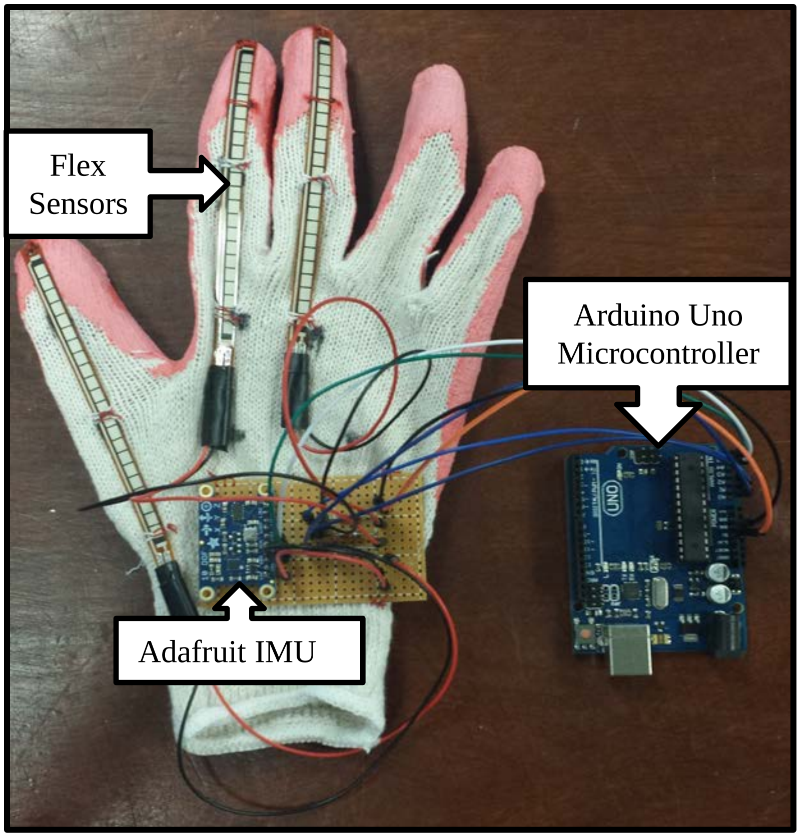


Figure 8. Fully assembled glove controller with flex sensors, IMU and Arduino Uno Microcontroller.

1. Inertial Measurement Unit

For the IMU, we chose Adafruit's 10 DOF breakout board shown in Figure 9. Adafruit's website states, "the board captures ten distinct types of motion or orientation related data by combining a 3-axis accelerometer, a 3-axis gyroscope and a barometric pressure sensor to provide stable and reliable readings" and "when the accelerometer and gyroscope are paired, they can be used as an inertial guidance system or 3D motion capture" [8]. The right top of the breakout board has the orientation of x, y and z and the corresponding Euler angles references. Euler angles describe orientation (in degrees) around a single reference point in three-dimensional space [1].

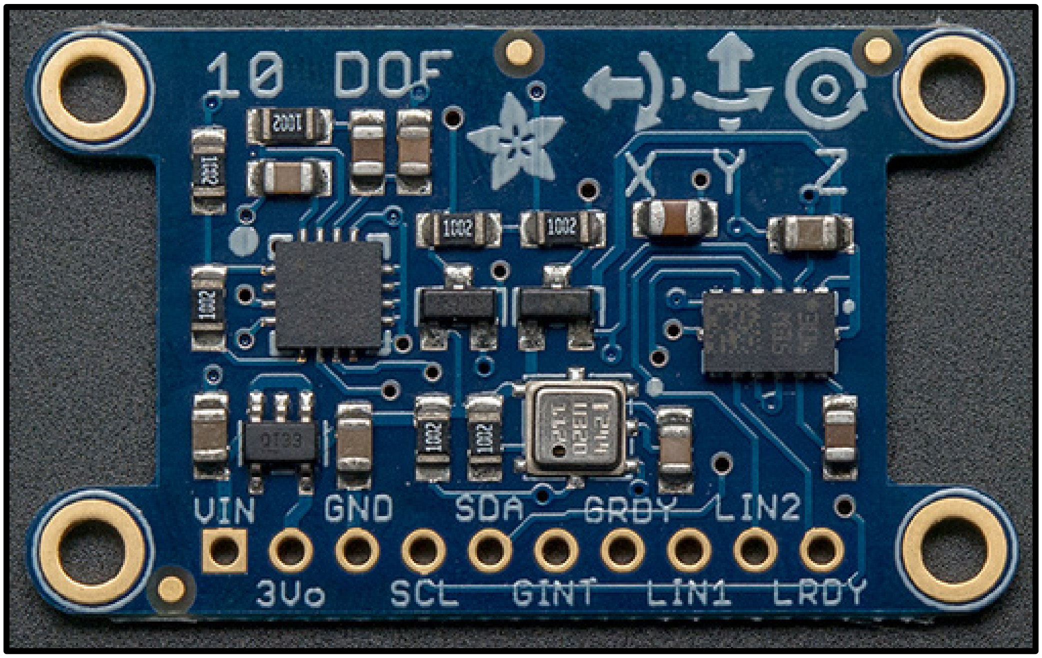


Figure 9. IMU containing an accelerometer, gyroscope and barometric pressure sensors from [9] used to get the pitch, roll and yaw to control the JACO arm.

2. Microcontroller

The Arduino Uno microcontroller controls raw data from the sensors, contains open source code, is small and has easy-to-use software. The Uno has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button as can be seen in Figure 10 from [9]. For Prototyping, the Uno is connected to a laptop via a USB connection. For Operations, the program is loaded to the Uno in stand-alone mode. Here we provide external power and communicate via a Wi-Fi shield.

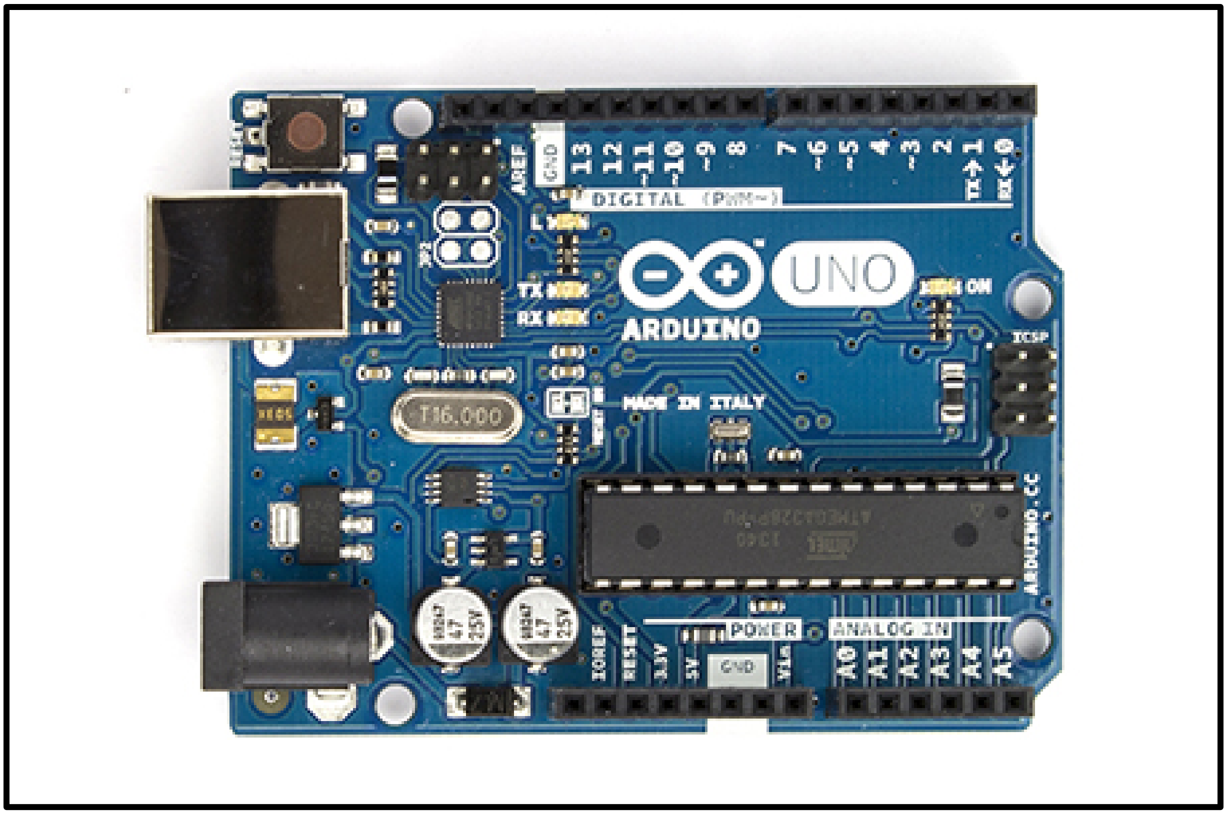


Figure 10. Arduino Uno from [9] used on the glove controller to collect analog data from the flex sensors and I2C from the IMU to send to the JACO Arm.

3. Flex Sensors

Three 4.5 inch Spectra Symbol flex sensors, are used for end effector control, see Figure 11. The sensors consist of conductive and resistive ink sections on a Kapton substrate [10]. The flex sensor has a flat resistance ranging 7–13 $k\Omega$ and a maximum resistance of two times the flat resistance for a 180° pinch bend [10]. One was used for the thumb, and the other two were for the first two fingers. The flex sensors were sewn in place at each knuckle of its corresponding finger to ensure that they would bend with the finger.

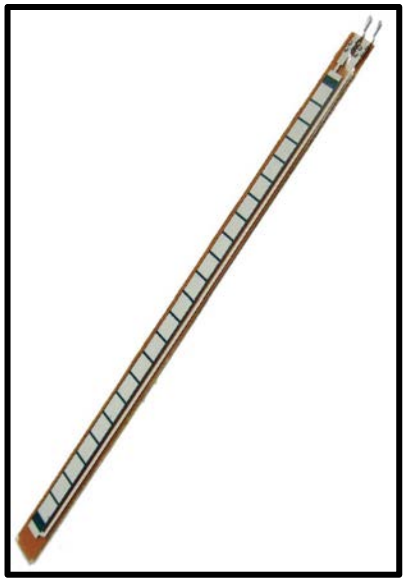


Figure 11. Flex sensorused on the glove controller to operate the end effectors on the JACO Arm, from [10].

4. Assembly

In Figure 12, a breadboard is used to hold the IMU, resistors and connections. All components were soldered onto the breadboard ensuring contacts would not break with movement. We strengthened the links with the flex sensors by soldering the wires to the connectors, heat shrinking and electrical taping each connector as recommended in [7].

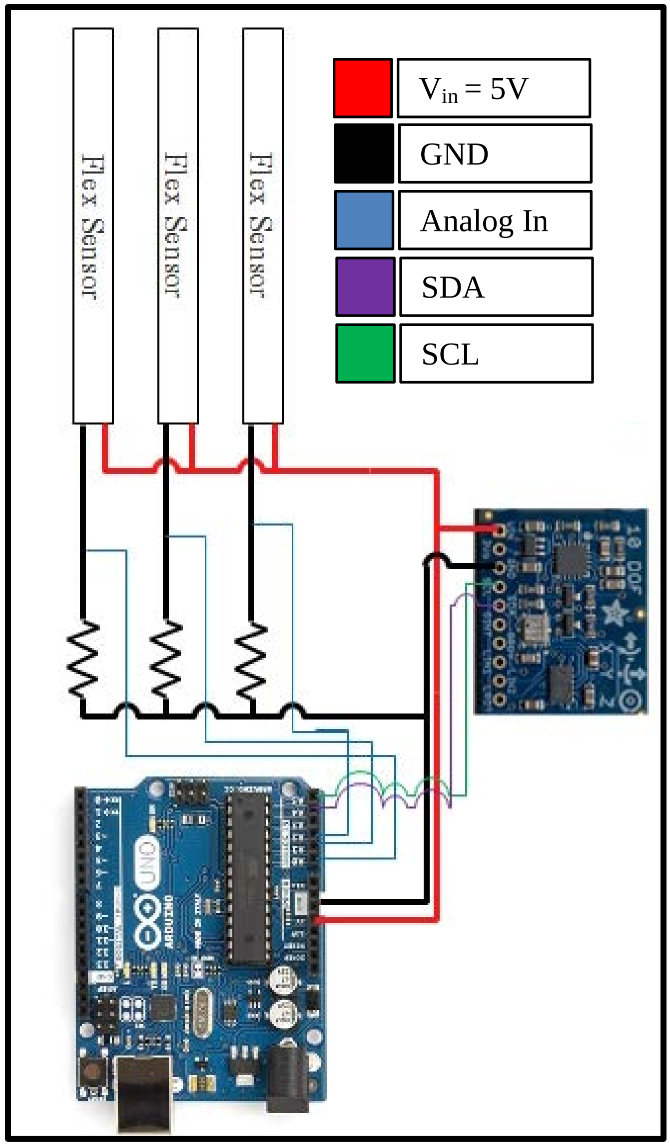


Figure 12. Electrical schematic of fully assembled glove controller.

C. COMMUNICATION INFRASTRUCTURE

We chose ROS as the operating system to support tele-optic control of the JACO arm. It has been around for many years and has a vast library of open source code to use for programming. ROS has a collection of libraries to make writing robot software easier [11]. ROS allows users to create packages (software). Within the package, the user can create nodes which are functions that perform computations [11]. Nodes talk to each

other via Topics. Topics are zones that transport data and to which nodes can subscribe or publish [11]. ROS simplifies the infrastructure process and makes it easier for the programmer. Kinova has a primary ROS package for customers to configure their JACO arm as needed.

IV. OPERATION

A. CONTROLLER

The Uno microcontroller is the brains for the operation of the glove controller. It receives the raw data from the flex sensors and IMU and sends the signal via ROS to the JACO arm.

For the flex sensors, as the user's fingers close, the resistance changes thereby regulating output voltage sent to ports A0, A1 and A2 of the Uno microcontroller. The microcontroller sends a signal to the JACO arm to shut the end effector. When the user brings their fingers back to natural positions, the end effector will reopen. Output voltage fluctuates with flex sensor varying resistance per Equation 10.

$$V_{out} = V_{in}\left(\frac{R_2}{R_1 + R_2}\right); \; R_1 \equiv \text{ Flex Sensor Resistance} \qquad (11)$$
$$R_2 = 22\text{k}\Omega$$

Figure 13 is the flex sensor's electrical diagram showing the regulation of 5V for output voltage to the Uno microcontroller. The microcontroller will see voltages ranging from 3.79 V to 2.29 V depending on the initial flat resistance and bend.

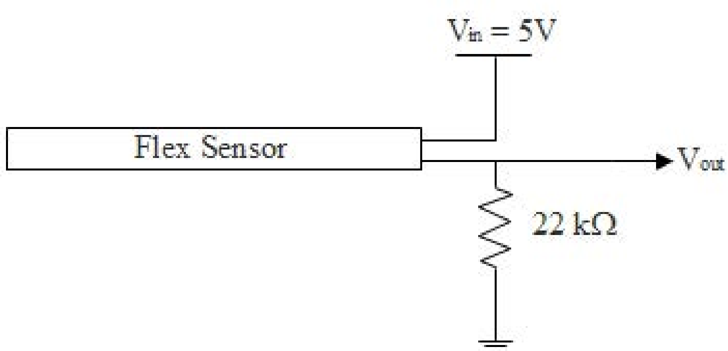


Figure 13. Voltage divider, for each flex sensor, which supplies analog voltage to the Arduino Uno microcontroller.

For the IMU, the Uno microcontroller uses an attitude and heading reference system (AHRS) sketch to read raw data from the board's accelerometer/magnetometer and convert it into Euler angles [1]. Therefore, the user's hand position provides a three-dimensional orientation data stream in terms of Roll (x), Pitch (y) and Yaw (z) in degrees. The data is sent to the JACO arm via ROS. The Uno microcontroller is a node in the JACO package.

B. TRAJECTORY CONTROL

The JACO arm contains 6 actuators and three fingers. The orientation signals from the glove are passed to the actuators via ROS to energize the actuators that move the manipulator. ROS reads the Uno microcontroller as a node, in the JACO ROS package, and then publishes to a topic. The user only controls movements of the end effector [12]. Kinova uses proprietary algorithms to transform DH parameters to JACO physical angles to pilot the different joints automatically [12]. The manipulator has joint and speed limits for protection. The user is capable of adjusting the speed of the manipulator as long as it stays below the max of 20 cm/s.

V. RESULTS AND CONCLUSIONS

JACO arm testing and glove experiments were completed. The JACO software was installed and verified for standalone operations within its range of motion. Testing of ROS and glove/manipulator interaction proved ineffective. Successful testing of the sensors on the glove confirmed regular hand movements' simulation.

A. RESULTS

To verify the glove controller functioned correctly, it was tested using the Arduino and visually confirmed with the 3D simulator Processing software. The coding for the glove controller is in Appendix A.

The flex sensors were tested first. Data was collected in Table 1 showing the results when the user contracts their fingers (thumb, index finger and middle finger) and then returning them to their normal positions. In the coding, the voltage output was mapped to an angle roughly proportional to the bend angle of the user's fingers.

Table 1. Flex sensors datawhile contracting and subsequent release of user's fingers. The bend angles are analog data converted from the output voltage, using Equation 11, in the Arduino code.

Bend Angles (deg)		
Thumb	**Index Finger**	**Middle Finger**
34	37	31
40	81	63
40	106	87
49	131	100
63	144	114
75	167	117
111	170	146
124	151	149
142	139	163
143	127	156
150	117	153
94	102	115
49	91	83
40	69	73
39	16	17

This data was then plotted in Figure 14, showing how the angle increases while the fingers contract and decreases back to original reading when the fingers are released. Future coding will be developed having the JACO arm close the end effector when the bend angle reaches 60^{o} and reopens when it returns below 50^{o}. The fingers will stop closing on an object when the force reaches 7N per default programming [12].

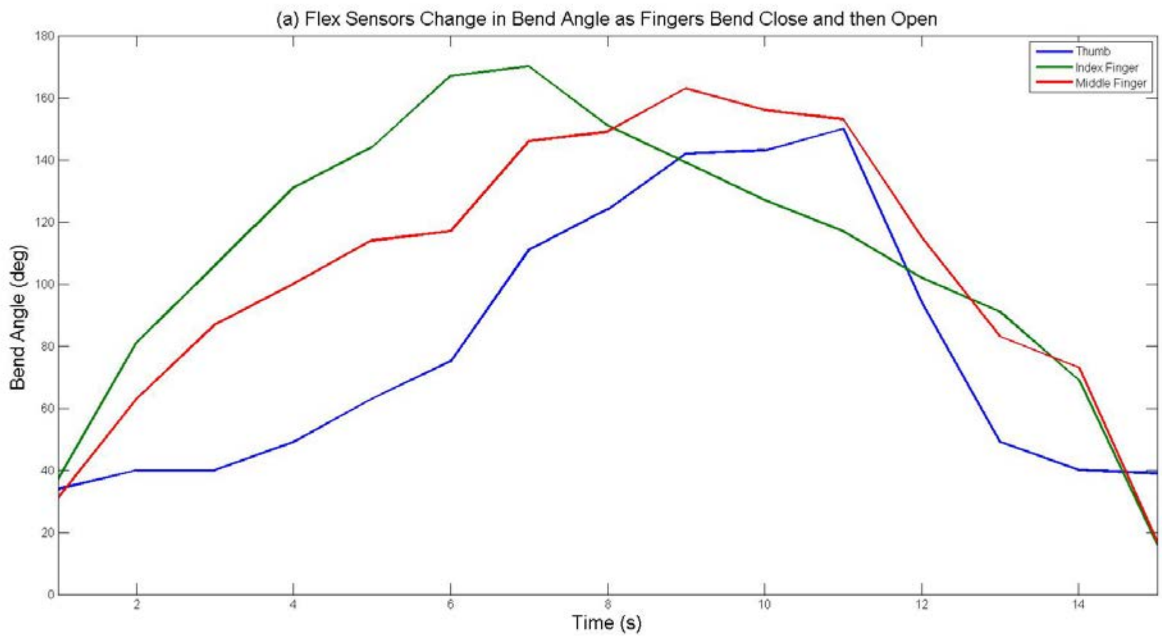
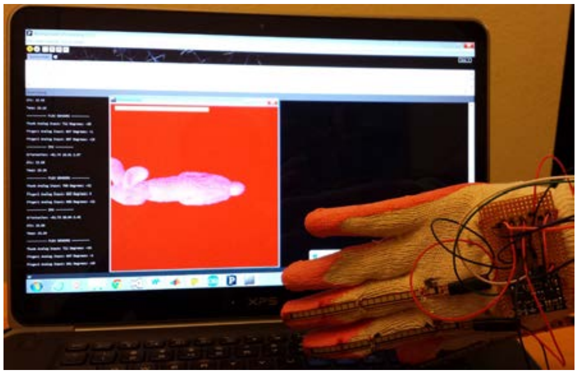
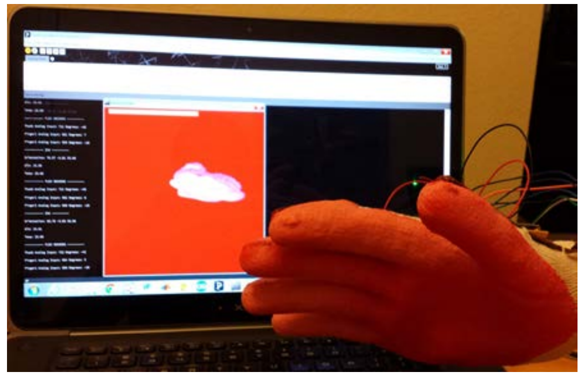
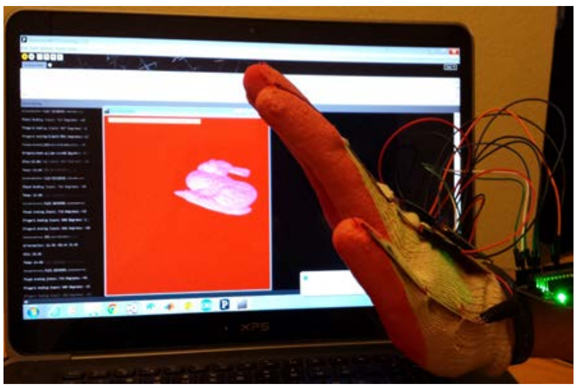


Figure 14. Plots for flex sensors datawhile contracting and subsequent release of user’s fingers. The plot shows the coding works allowing us to use the output to control the end effector.

The IMU was tested using an existing example sketch from the Processing software to display IMU operation. The user changes the orientation of the object based on the position of the glove. The sketch receives 3D orientation data from the LSM303DLHC 3-axis accelerometer and displays it via the 3D simulator; see Figures 15(a) – 15(d).


(a)


(b)


(c)

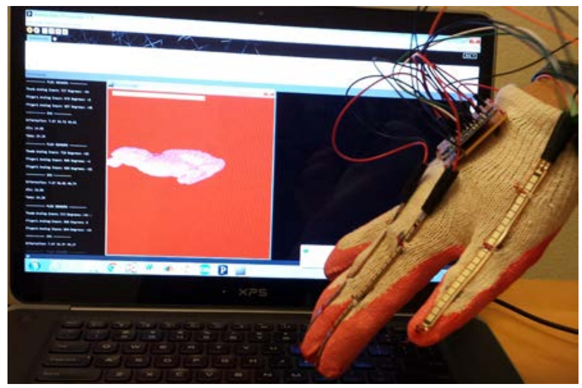

(d)

Figure 15. Simulation of IMU output from glove controller. Figures 14(a)–14(d) show a 3D object, a rabbit in this simulation, orient itself in the same orientation of the user’s hand.

B. CONCLUSION

The JACO arm works as desired with smooth operation and the ability to vary the speed of operation. The JACOSOFT software allowed us to become familiar with the operation of the manipulator both with the joystick and trajectory mapping. The Health Monitor for the JACO arm displayed the force, current and position of each joint and finger allowing us to become familiar with the parameters.

ROS did not successfully communicate with the JACO arm. The driver in the JACO-ROS package was faulty and needs to be further investigated. An alternate choice for communication would be the Visual Studio JACO API written by Kinova. Developer examples are available.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

This sketch was a combination of example sketches from [1] and [7]. I made modifications to the sketch for our design and used it in the Processing Software 3D simulator to verify proper IMU and flex sensors operation.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_BMP085_U.h>
#include <Adafruit_Simple_AHRS.h>

// Create sensor instances.
Adafruit_LSM303_Accel_Unified accel(30301);
Adafruit_LSM303_Mag_Unified   mag(30302);
Adafruit_BMP085_Unified       bmp(18001);

// Create simple AHRS algorithm using the above sensors.
Adafruit_Simple_AHRS          ahrs(&accel, &mag);

// Update this with the correct SLP for accurate altitude measurements
float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;

void setup()
{
  Serial.begin(115200);
  Serial.println(F("Adafruit 10 DOF Board AHRS Example")); Serial.println("");

  // Initialize the sensors.
  accel.begin();
  mag.begin();
  bmp.begin();
}

void loop(void)
{
  sensors_vec_t   orientation;

  // Use the simple AHRS function to get the current orientation.
  Serial.println("----------- IMU ----------");
  if (ahrs.getOrientation(&orientation))
  {
    /* 'orientation' should have valid .roll and .pitch fields */
```

```
  Serial.print(F("Orientation: "));
  Serial.print(orientation.roll);
  Serial.print(F(" "));
  Serial.print(orientation.pitch);
  Serial.print(F(" "));
  Serial.print(orientation.heading);
  Serial.println(F(""));
}

// Calculate the altitude using the barometric pressure sensor
sensors_event_t bmp_event;
bmp.getEvent(&bmp_event);
if (bmp_event.pressure)
{
  /* Get ambient temperature in C */
  float temperature;
  bmp.getTemperature(&temperature);
  /* Convert atmospheric pressure, SLP and temp to altitude */
  Serial.print(F("Alt: "));
  Serial.print(bmp.pressureToAltitude(seaLevelPressure,
                                      bmp_event.pressure,
                                      temperature));
  Serial.println(F(""));
  /* Display the temperature */
  Serial.print(F("Temp: "));
  Serial.print(temperature);
  Serial.println(F(""));
}

//Defines analog input variables
int thumb, degrees1;
int finger1, degrees2;
int finger2, degrees3;

//Read the voltage from the voltage divider (sensor plus resistor)
thumb = analogRead(0);
finger1 = analogRead(1);
finger2 = analogRead(2);

// convert the voltage reading to degrees
degrees1 = map(thumb, 694, 575, 0, 180);
degrees2 = map(finger1, 666, 548, 0, 180);
degrees3 = map(finger2, 679, 573, 0, 180);

// print out the result
```

```
  Serial.println("----------- FLEX SENSORS ----------");
  Serial.print("Thumb Analog Input: "); Serial.print(thumb,DEC);
  Serial.print(" Degrees: "); Serial.println(degrees1,DEC);
  Serial.print("Finger1 Analog Input: "); Serial.print(finger1,DEC);
  Serial.print(" Degrees: "); Serial.println(degrees2,DEC);
  Serial.print("Finger2 Analog Input: "); Serial.print(finger2,DEC);
  Serial.print(" Degrees: "); Serial.println(degrees3,DEC);

  delay(100);
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Adafruit. (2014, Mar. 19). AHRS for Adafruit's 9-DOF, 10-DOF, LSM9DSO Breakouts. [Online]. Available: https://learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout

[2] Euler Angles. (2015, May 06). *Wikipedia*. Available: http://en.wikipedia.org/wiki/Euler_angles. Accessed May 09, 2015.

[3] Dynamics (mechanics). (2015, May 06). *Wikipedia*. Available: http://en.wikipedia.org/wiki/Dynamics_(mechanics). Accessed May 11, 2015.

[4] H. Choset et al., *Principles of Robot Motion*, 1^{st} ed. Cambridge, MA: The MIT Press, 2005, pp. 349–370.

[5] R. D. Klafter, T. A. Chmielewski and M. Negin, *Robotic Engineering: An Integrated Approach*, 1^{st} ed. Englewood Cliffs, NJ: Prentice Hall, 1989, pp. 615–623.

[6] JACO Specifications. (2014). Kinova. [Online]. Available: http://kinovarobotics.com/wp-content/uploads/2015/01/JACO-2-Specification-Sheet-1.0.1.pdf

[7] D. S. Churman. (n.d.). DIY robotic hand controlled by a glove and Arduino. [Online]. Available: http://www.instructables.com/id/DIY-Robotic-Hand-Controlled-by-a-Glove-and-Arduino/?ALLSTEPS

[8] Adafruit. (2013, Nov. 29). Adafruit 10-DOF IMU breakout. [Online]. Available: https://learn.adafruit.com/adafruit-10-dof-imu-breakout-lsm303-l3gd20-bmp180

[9] Arduino. (2015). Arduino Uno. [Online]. Available: http://www.arduino.cc/en/Main/ArduinoBoardUno

[10] Flex Sensor. (n.d.). SpectraSymbol. [Online]. Available: http://www.spectrasymbol.com/wp-content/themes/spectra/images/datasheets/FlexSensor.pdf

[11] About ROS. (n.d.). ROS. [Online]. Available: http://www.ros.org/about-ros/

[12] *JACO Research Edition User Guide*, V1.0.0, Kinova, Boisbriand, Canada, 2011–2012, pp. 18–19.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California